5. Cipuri UCP

Performantele si capacitatea unui calculator (PC în particular) sunt determinate, în primul rând, de microprocesor, componenta-cheie a oricarui sistem de calcul. În prezent, exista peste 50 de microprocesoare compatibile Intel, având la baza arhitectura x86, inclusiv unele procesoare RISC.

Avantajul incontestabil al microprocesoarelor Intel x86 si al clonelor produse de AMD, Cyrix si NexGen, îl constituie mentinerea compatibilitatii cu modelele anterioare; astfel, cel mai performant Pentium poate executa software scris pentru x86.

5.1. Microprocesoarele INTEL X86

Procesoarele Intel x86, care intra în categoria procesoarelor CISC, sunt cele mai raspândite procesoare în momentul actual. Primul procesor din seria x86, Intel 8086 apare în anul 1978. Acesta avea registre pe 16 biti, o magistrala extema tot pe 16 biti si o adresare pe 20 biti, ceea ce permitea un spatiu de adresa de 1 MB. Intel 80286 introduce modul de lucru protejat, mod în care registrele de segment erau privite ca selectori sau *pointeri* în tabele de descriptori. Descriptorii de segment furnizau o adresa pe 24 de biti, ceea ce însemna un spatiu de adresa de 16 MB. De asemenea, tot 80286 a introdus suport pentru memorie virtuala, bazata pe *swaping* de segmente, si diferite mecanisme de protectie - verificarea limitei segmentelor, segmente de tip *read-only* sau *execute-only* si 4 niveluri de protectie la dispozitia sistemului de operare.

Intel 80386 aduce ca noutate registre pe 32 de biti, atât pentru operanzi, cât si pentru calculul adreselor. Totusi, registrele de 32 de biti pot fi accesate si ca registre pe 16 biti pentru compatibilitatea cu versiuni anterioare. A fost introdus un nou mod de lucru - modul virtual - pentru a obtine o elicienta sporita la executia programelor create pentru 8086. Adresarea pe 32 de biti permite un spatiu de adresa de 4 GB si, de asemenea, permite ca fiecare segment sa poata avea 4 GB. Ca noutate, pe lânga segmentare, se introduce si un mecanism de paginare cu pagini de dimensiune fixa de 4 KB.

Procesorul Intel 80486, realizat în tehnica benzii de asamblare, integreaza o memorie cache de nivel 1 cu capacitatea de 8 KB, iar unitatea de lucru cu numere în virgula mobila este inclusa pe cipul procesorului.

Pentium este primul procesor superscalar produs de Intel. Acesta poate executa 2 instructiuni în fiecare ciclu. Memoria cache de nivel 1 este dublata fata de 80486, existand acum 8 KB pentru instructiuni si 8 KB pentru date. Registrele pe 32 de biti stint mentinute, dar sunt adaugate cai de date interne pe 128 si 256 biti, pentru a mari performantele la transferul datelor.

Urmatorul procesor din serie este PentiumPro. Este tot un procesor superscalar, dar spre deosebire de predecesorul sau, acesta poate executa trei instructiuni pe ciclu. Sunt introduse toate tehnicile moderne de prelucrare a instructiunilor: executie *out-or-order*, predictia salturilor si executie speculativa etc. Recunoscând avantajele procesoarelor RISC, arhitectii Pentium Pro introduc trei decodificatoare, care au rolul de a transforma instructiunilc CISC în instructiuni RISC. Deci, cu toate ca PentiumPro este privit ca un procesor CISC, nucleul sau este de tip RISC. Pentru rnentinerea secventialitatii f1uxului de instructiuni exista un buffer de reordonare a instructiunilor, de unde instructiunile sunt retrase (deci pot modifica starea procesorului) în ordinea lor secventiala.

Pentium II adauga o extensie la setul clasic de instructiuni: instructiunile MMX, destinate special aplicatiilor multimedia, instructiuni care opereaza pe principiul SIMD (single instruction-multiple data). Memoria cache de nivel 1 este marita la 16 KB, pentru date si 16 KB pentru instructiuni, iar memoria cache de nivel 2 poate atinge capacitatea de 2 MB.

Pentium III se bazeaza pe arhitectura PentiumPro si Pentium II, dar introduce 70 de noi instructiuni si o noua unitate SIMD in virgula mobila. Aceasta extensie a setului de instructiuni poarta denumirea SSE *(Streaming SIMD Internet Extension).*

În tabelul 2.1. este prezentata în mod succint evolutia procesoarelor Intel x86.

5.1.1. Structura interna de baza

Dupa cum se observa din fig. 2.1, microprocesorul include doua componente majore: unitatea de executie si unitatea de interfata cu magistrala.

Unitatea de interfata cu magistrala executa toate ciclurile de magistrala *(read, write, întrerupere*), fie la

Procesorul	Data apariţiei	Performanţa (MIPS)	Frecvenţa	Numărul de tranzistori	Dimensiunea registrelor (biţi)	Spaţiul de adresă	Memoria cache
8086	1978	0,8	8 MHz	29 K	16	1 MB	-
80286	1982	2,7	12,5 MHz	134 K	16	16 MB	-
80386	1985	6,0	20 MHz	275 K	32	4 GB	-
80486	1989	20	25 MHz	1,2 M	32	4 GB	8 K L1
Pentium	1993	100	60 MHz	3,1 M	32	4 GB	16 KB L1
Pentium Pro	1995	440	200 MHz	5,5 M	32	64 GB	16 KB L1 512 KB L2
Pentium II	1997	466	266 MHz	7 M	32	64 GB	32 KB L1 2 MB L2
Pentium III	1999	1000	500 MHz	8,2 M	32 reg. generale 128 reg. SIMD FP	64 GB	32 KB L1 2 MB L2

Tabelul 5.1. Evolutia procesoarelor Intel

cererea unitatii de executie, fie pentru umplerea cozii de instructiuni (coada de instructiuni este un registru FIFO alcatuit din 6 cuvinte, instructiunile asteptând aici intrarea în executie). Ciclurile de încarcare ale instructiunilor sunt executate în intervalele de timp în care unitatea de executie nu solicita magistrala. Daca unitatea de executie nu solicita magistrala si coada de instructiuni este plina, atunci au loc cicluri inactive pe magistrala.
Executia instructiunilor de salt duce la resetarea cozii, deoarece trebuie extrase instructiuni din alta zona de memorie.

Unitatea de executie obtine instructiuni de la unitatea de interfata cu magistrala (în cazul în care coada este vida, asteapta) si le executa lucrând cu adrese si date. În functie de rezultatul fiecarei operatii executate, actualizeaza registrul de flags. Dupa executia instructiunii, furnizeaza date si adrese catre unitatea de interfata cu magistrala. O alta activitate desfasurata de unitatea de executie, o constituie calcularea adreselor efective ale operanzilor, conform modului de adresare, prin *relocarea* adresei efective.

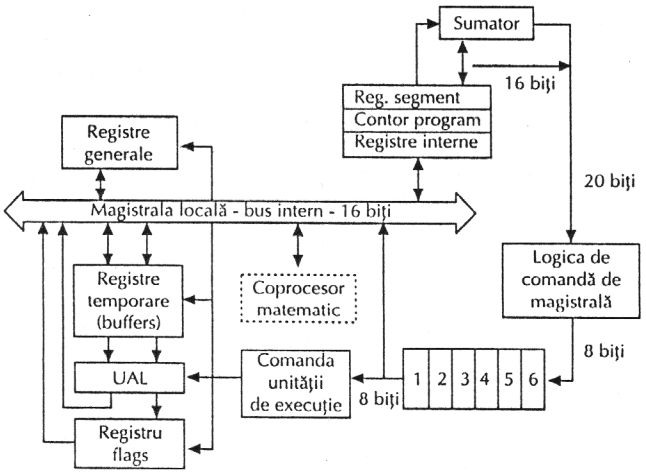
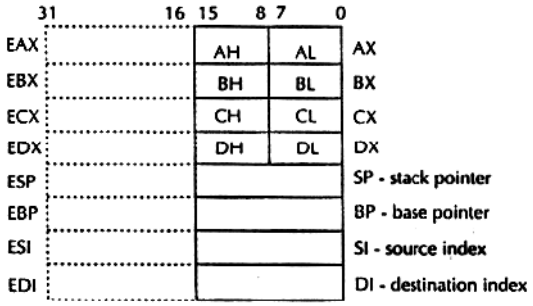


Figura 5.1. Structura interna de baza a microprocesoarelor Intel

Fiecare **ciclu de magistrala** consta în intervale de timp date de semnalul de tact, suplimentate cu un timp de asteptare, daca semnalul nu este *Ready;* pe durata ciclurilor inactive, se realizeaza demultiplexarea prin generarea starii anterioare în *latch-uri* (CBB-RS) exteme, pe frontul scazator al impulsului de tact.

5.1.2. Registrele generale si stiva

Registrele generale cuprind:
1) registre de utilizare generala. AX, BX, CX, DX adresabile direct pe 16 biti, fiecare putând servi ca destinatie a datelor (acumulator); registrele pot fi adresate si pe *bytes,* prin specificarea byte-ului inferior (AL-*ALow*) sau superior *(AH-AHigh)* corespunzatoare bitilor 0-7 (L), respectiv 8-15 (H); începând cu microprocesoarele 180386 si pâna la Pentium, acestea se pot adresa si pe dublu cuvânt (32 de biti) prin specificarea registrului precedat de E *(Extended):* EAX, BOX, BCX, EOX. Registrele de ulilizare generala au fost proiectate sa aiba o deslinatie specifica, în concordanta cu operatiile pe care le executa:



(E - *Extended* - pentru microprocesoarele 180386 si ulterioare, care lucreaza pe 32 de biti.)

Figura 5.2. Registrele generale

AX - este utilizat pentru operatiile de înmultire si împartire, pe 16 biti, respectiv pentru operatii de intrare/iesire pe 16 biti;

AL - este utilizat, pentru aceleasi operatii ca si AX dar pe 8 biti; în plus se utilizeaza pentru operatii BCO *(binary coded decimal)* si conversii de cod;

AH - este folosit pentru înmultire si împartire pe 8 biti;

BX - se utilizeaza în conversii de cod si ca registru de baza la adresare;

CX - are rol de contor de ciclu în cazul structurilor repetitive cu incrementare, utilizat fiind si în operatiile cu siruri;

DX - este utilizat ca registru de adresare indirecta la porturile de intrare/iesire, precum si la operatiile de înmultire/împartire.

2) **registrul indicator de stiva** de 16 biti, continând adresa varfului stivei (**SP**- *Stack Pointer);* este utilizat implicit în toate operatiile cu stiva.

3) **registrul indicator de baza** de 16 biti, contine adresa de baza **(BP** - *Base Pointer).*

4**) registrele de index SI** *(Source Index)* si **DI** *(Destination Index)* de 16 biti participa la elaborarea adreselor, fiecare adresa rezultând prin însumarea diferitelor combinatii dintre adresa de baza, un indice (index) si o deplasare (offset).

Stiva este o lista liniara la care orice acces (extragerea sau inscrierea unui element) se face numai la unul din capete; în cazul stivei, se manipuleaza ultima data din lista, de aceea stiva se mai numeste si lista LIFO *(Last In - First Out)*.

Elementele se introduc/extrag la/de la începutul sau vârful stivei *(top of stack).* Sfârsitul stivei este ultimul accesibil si nu va putea fi extras decât dupa ce toate celelalte elemente au fost extrase. Adresa elementului din vârful stivei se afla în registrul SP *(Stack Pointer).*

În figura 5.3. este reprezentat un segment de stiva si registrele implicate în operatia de adresare a unei stive. Registrul BP nu va puncta automat baza stivei. De regula, dupa o încarcare prealabila de catre programator cu o anumita valoare, de exemplu, cea de la baza stivei, BP va fi întrebuintat pentru a avea acces la diferiti parametri plasati în stiva.

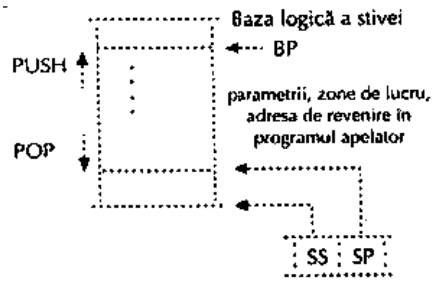


Figura 5.3. Adresarea unei stive

Nu trebuie confundata baza logica a stivei cu adresa de baza a segmentului de stiva, aflata în registrul de segment **SS** *(Stack Segment).*

Instructiunile PUSH si POP vor afecta continutul registrului SP, nu si al registrelor SS si BP. Instructiunile CALL si RET afecteaza, de asemenea, continutul registrului SP, dar pot afecta si registrul BP uneori.

Vom clarifica printr-un exemplu mecanismul de adresare al stivei. Fie codul urmator:

```
PUSH AX        ; depun în stiva continutul registrului AX
POP CX         ; pe care apoi îl aduc in registrul CX
POP DX         ; în DX preiau 2 octeti din stiva
MOV BP, SP     ; dedublez continutul registrului SP in BP
```

Presupun situatia din figura 5.4.a. drept situatia initiala.

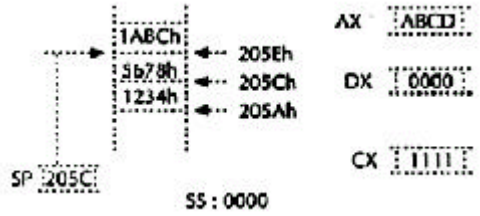


Figura 5.4.a. Situatia initiala a registrelor si stivei

Dupa executarea instructiunii PUSH AX, continutul lui AX se va depune în stiva, în cuvântul punctat de la adresa relativa 205Ch, peste valoarea 5678h existenta acolo (vezi fig. 5.4.b.) apoi registrul SP va contine valoarea 205Ah.

Figura 5.4.a. Situatia registrelor si stivei dupa PUSH AX

Prima instructiune POP va conduce la marirea mai întâi a continutului lui SP cu 2, urmând apoi extragerea cuvântului ABCDh din stiva. Ca atare, continutul registrului CX va deveni egal cu ABCDh. A doua instructiune POP mareste din nou cu 2 continutul lui SP si, ca urmare, în registrul DX va fi încarcata valoarea 1ABCh, aflata în stiva. Dupa executarea instructunii MOV BP, SP, registrul BP va avea acelasi contnut, 205Eh ca si al registrului SP.

La sfârsitul operatiilor, registrele si stiva vor arata ca în figura 5.4.c.

Figura 5.4.c. Situatia finala a registrelor si stivei

5.1.3. Registrul indicatorilor de conditie (reg. Flags)

Indicatorii de conditie sunt utilizati pentru a memora informatii referitoare la rezultatul unor operatii aritmetice sau logice (AF, CF, OF, PF, SF, ZF) si pentru memorarea unor informatii de control pentru microprocesor (DF, IF, TF). Aspectul acestui registru este figurat in figura 5.5.

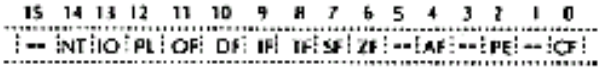


Figura 5.5. Registrul indicatorilor de conditie

Semnificatiile acestor indicatotori sunt urmatoarele:

(I) - Indicatorul CF *(Carry Flag),* bitul de transport, este 1, daca în executia unei instructiuni care pozitioneaza acest indicator a aparut un transport, sau s-a facut un împrumut la rangul cel mai semnificativ. De asemenea, instructiunile de rotire a continutului unui registru pot sa pozitioneze acest indicator;

(2) - Indicatorul PF *(Parity Flag),* bitul de paritate, este 1, daca din executia unei instructiuni care pozitioneaza acest indicator s-a obtinut un rezultat numar par de biti cu valoarea 1;

(3) - Indicalorul AF *(Auxiliary Carry Flag),* bitul pentru transporlul auxiliar, este 1, daca în executia unei instructiuni care pozitioneaza acest indicator a aparut un transport de la rangul 3 spre rangul 4, sau a fost execluat un împrumut din rangul 4 spre rangul 3. Acesl indicator esle utilizat pentru implementarea aritmeticii zecimale codificate binar;

(4) -Indicatorul ZF (Zero *Flag),* bitul indicator al rezultatului nul, este 1, daca în executia unei instructiuni care pozitioneaza acest indicator, s-a obtinut rezultatul zero;

(5) - Indicalorul SF *(Sign Flag),* bitul de semn, este 1, daca din executia unei instructiuni care pozitioneaza acest indicator s-a obtinut un rezultat pentru care bitul cel mai semnificaliv este 1 (rezultat negativ);

(6) - Indicalorul TF *(Trace Flag),* bitul de depanare, este ulilizat pentru conlrolul executiei instructiunilor în regim pas cu pas, în scopul depanarii programelor. Daca acest indicator este 1, dupa executia fiecarei instructiuni se va genera un semnal de întrerupere intern (pe nivel 1);

(7) - Indicalorul IF *(Interrupt Flag),* bitul de activare/dezactivare a sistemului de întreruperi, controleaza acceptarea semnalelor de întrerupere externa. Daca indicatorul IF esle 1, este activata acceptarea semnalelor de întrerupere externa. Indicatorul nu are influenta în cazul semnalului de întrerupere nemascabila;

(8) - Indicalorul DF *(Direction Flag),* bitul de "directie", indica directia de parcurgere a sirurilor de octeti în cazul instructiunilor pe siruri de octeti. Valoarea zero a acestui indicator indica parcurgerea sirurilor de la adrese mici spre adrese mai mari;

(9) - Indicatorul OF *(Overflow Flag),* bitul de depasire în virgula fixa, este 1, daca din executia unei instructiuni aritmetice cu semn a aparut o depasire, adica s-a obtinut un rezultat care nu poate sa fie memorat corect în destinatia stabilita de catre instructiune.

O data cu aparitia microprocesoarelor I80286, ce permit executia mai multor taskuri, registrul de flags a fost completat cu bitii:

(12-13) – indicatorul IOPL *(Input Output Privilege Level),* indicator al nivelului privilegiat de acces la operatiile de intrare-iesire;

(14) indicalorul NT *(Nesled Task* -task imbricat), ce permite revenirea la task- ul care l-a operat în mod normal (NT=0) sau printr-o instructiune de revenire IRET (NT= 1).

La microprocesoarele pe 32 de biti, regislrul de flags (EFLags) are structura (fig. 5.6):

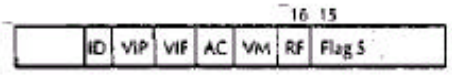


Figura 5.6. Structura registrului indicatorilor de conditie
La microprocesoarele pe 32 de biti

RF *(Resume Flag)* - indicator de rezumat al bitilor setati în registrul indicatorilor, util pentru depanarea programelor;

VM *(Virtual Mode)* - indicator de comutare în mod real/virtual;

AC *(Alignment Check)* – disponibil la procesoarele I80486 si Pentium. Controlul alinierii este setat atunci când referirile la adresele de memorie nu se gasesc în intervalul de adrese atribuit în scopul protectiei.

La microprocesoarele Pentium, exista în plus trei flaguri:

VIF *(Virtual Interrupt Flag)* - indicator de întreruperi virtuale;

VIP *(Virtual Interrupt Pending)* - indicator de întreruperi virtuale în asteptare;

ID *(Identification)* - determina pe ce clasa de procesor urmeaza executia. Mecanismul de depistare al procesorului se desfasoara astfel:

a) daca programul ce se executa pe 16 biti descopera ca exista biti care asteapta setarea în afara celor 16 biti si registrului de indicatori, înseamna ca executia se realizeaza pe cel putin un microprocesor I80386;

b) se testeaza capacitatea de sesizare a violarii spatiului de adrese alocat, care se putea seta AC la I80486 sau Pentium;

c) daca programul poate scrie ID, atunci microprocesorul este Pentium.

La MS-DOS, prin lansarea în executie a depanatorului *DEBUG* urmat de comanda R, se va afisa continutul registrelor si configuratia indicatorilor de conditie (iesirea se va face cu comanda *Quit).*

5.1.4. Întreruperi si exceptii

Capacitatea unui PC de a suspenda o activitate pe care o executa la un moment dat si a comuta pe o alta activitate constituie reactia sistemului la solicitarea unei întreruperi.

Microprocesoarele dispun de capacitatea de a fi întrerupte, combinata cu posibilitatea de a conserva activitatea întrerupta pe parcursul prelucrarii solicitate prin întrerupere. În acest scop, microprocesorul foloseste stiva în care va memora starea activitatii întrerupte. Dupa tratarea întreruperii va extrage din stiva starea activitatii întrerupte, pe care o va continua exact din locul în care a fost întrerupta.

Fiecare componenta a PC care solicita microprocesorul la un moment dat, are propriul nivel de intrerupere:

- tastatura;
- ceasul intern;
- discurile;
- imprimantele s.a.

Exista trei categorii majore de întreruperi:

- **întreruperi hardware** - care sunt cauzate de evenimente externe generate de componente ale PC (exemplele anterioare);
- **exceptiile** - sunt întreruperi software care rezulta dintr-un program în executie; ele apar atunci când o instructiune nu poate fi executata complet în mod normal; exemplu: împartirea unui numar la zero; sunt întreruperi nemascabile (nu pot fi inhibate);
- **întreruperi software**, care solicita anumite servicii prin instructiuni de tipul INT utilizate în programe; actiunile solicitate prin aceste întreruperi se regasesc sub forma unor functii din ROM-BIOS; sunt nemascabile.

Pentru localizarea cauzei întreruperii, microprocesorul asociaza fiecarei surse de întrerupere un numar, numit **vector de întrerupere**, memorat într-un tabel de descriere a întreruperilor (IDT - *Interrupt Description Table)* localizat la începutul memoriei RAM; vectorul de întrerupere contine adresa unde se gaseste rutina (programul) de tratare a întreruperii.

Modul de tratare a intreruperilor (fig. 5.7).

1. la un moment dat, microprocesorul executa un program, avand in registrul de flags bitul IF = 0 (nu este solicitata nici o întrerupere);
2. la urmatorul moment de timp (impuls de tact) este solicitata o întrerupere; daca aceasta nu este mascata, va atentiona microprocesorul prin setarea bitului IF din registrul de nags (IF=1) (daca ar fi mascata, IF ar ramâne o si microprocesorul ar ignora-o);
3. microprocesorul memoreaza starea programului întrerupt în stiva;
4. în functie de nivelul întreruperii, se gaseste în IDT vectorul de întrerupere care identifica adresa de memorie unde se afla rutina de tratare a întreruperii;
5. se executa rutina de întrerupere ca orice alt program;
6. la încheierea executiei rutinei de întrerupere se reface starea programului întrerupt, prin citirea din stiva a starii la care a fost întrerupt si se continua executia acestuia.

Observatie: întreruperile sunt semnalate microprocesorului prin semnale emise de catre controlerul de întreruperi.

Fig, 5.7. Tratarea întreruperilor

5.1.5. Registrele de segment si registrul contor de program

Registrele de segment permit microprocesorului sa adreseze direct memoria:

CS *(Code Segment)* pentru segmentul de program;

DS *(Data Segment)* pentru segmentul de date curent;
SS *(Stack Segment)* pentru segmentul de stiva;
ES *(Extra Segment)* pentru segmentul de date auxiliar;
FS, GS - segmente de date la microprocesoare ulterioare I 80386.
Aceste registre pot defini la un moment dat patru segmente de memorie de câte 64 K fiecare (64 K = 2^6. 2^{10}) în mod direct.

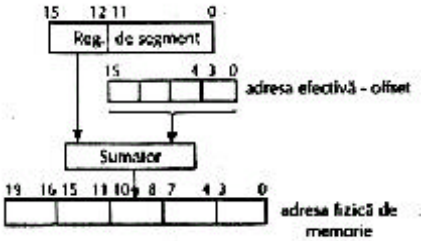


Fig. 5.8. Tehnica de segmentare

Adresarea în modul real permite microprocesorului sa acceseze un spatiu de adrese fizice de pâna la 1 M (2^{20}) prin tehnica de segmentare, tehnica ce nu adreseaza direct locatiile de memorie, ci printr-un procedeu care se desfasoara în doua etape:

- în prima etapa se încarca un registru de segment cu adresa unui bloc de memorie de 64 K, constituind adresa de baza a segmentului;
- la adresa fizica de baza a segmentului se aduna adresa de offset, care constituie deplasarea *(offset-ul)* relativa fata de adresa de baza în interiorul segmentului, rezultând adresa fizica de memorie.

Dezavantaj: programatorii trebuiau sa aiba grija, deoarece atunci când se depaseau limitele unui segment era necesara o noua reîncarcare a registrelor de segment.

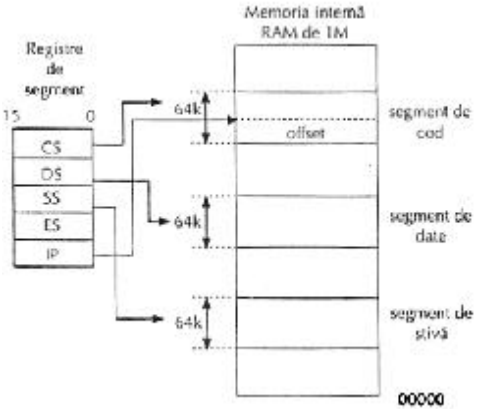


Fig. 5.9. Adresarea în modul real

Adresa relativa *(offset)* a instructiunii ce urmeaza a fi executata este pastrata în registrul contor de program **IP** *(Instruction Pointer);* pentru o instructiune de salt, continutul **IP** este salvat în vârful stivei si încarcat cu adresa relativa a instructiunii ce urmeaza a fi exec.utata din segmentuI de cod.

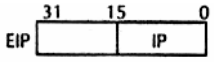


Fig. 5.10. Registrul contor de program

5.1.6. Adresarea memoriei în mod protejat

Segmentarea

Mecanismul de generare a adreselor în mod protejat este ilustrat in figura 5.11.

(1) Registrele de segment în acest mod de adresare nu contin adresa fizica de baza a segmentului, ci au rol de selectoare având structura:

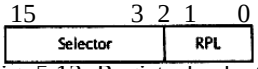


Fig. 5.12. Registrul selector

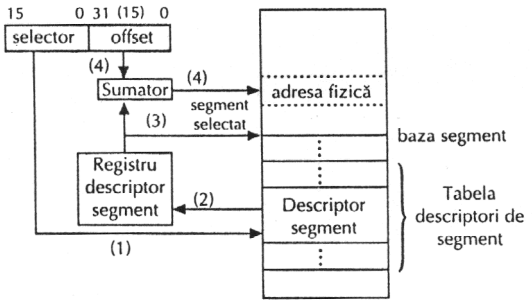


Fig. 5.11. Adresarea în mod protejat

RPL *(Register Privilege)* - indica nivelul privilegiat al registrului selector; acesta dispune de patru niveluri de protectie în functie de softul care solicita adresarea memoriei (fig. 5.12):

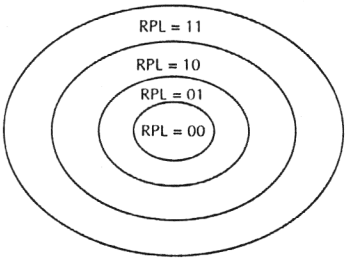


Fig. 5.13. Nivelurile de protectie

Protectia consta în capacitatea sistemului de a face ca erorile de software ale unui program sa nu afecteze alte programme. În acest scop, microprocesorul dispune de un mecanism de protectie privilegiat de tip inelar crescator, pe patru niveluri, în vederea izolarii aplicatiilor software de eventuale erori ale software-ului de baza.

Controlul comunicarii între programe si sistemul de operare este implementat prin separarea spatiului de adresa si mecanismul privilegiat. Controlul spatiului de adresa separa programele de aplicatie unele fata de altele, în limp ce mecanismul privilegiat izoleaza software-ul de baza de cel de aplicatie.

Protectia se bazeaza pe "ierarhia de tutela", organizata inelar pe cele 4 niveluri din figura 5.13:

- nivelul 0 - cea mai mare tutela,
- nivelul 3 - nivelul cu cea mai mica tutela.

De remarcat ca nivelul privilegiat este un atribut de protectie afectat în toate segmentele din software-ul de baza; acesta determina care proceduri pot accesa segmentul. Drepturile de acces si limitele sunt realizate prin hardware.

Codul sistemului de operare si segmentele de date plasate pe cel mai privilegiat nivel (nivelul 0) nu pot fi accesate direct prin programe situate pe alt nivel privilegiat; fiecare program poate accesa date de pe un nivel mai mare sau egal cu nivelul privilegiat care îi este asociat.

TI - indicatoruI tabelului de descriptori de segment.

Tabelele descriptorilor de segment defmesc toate segmentele utilizate, existând trei tipuri de tabele ale caror adrese sunt pastrate în registre dedicate:

- tabela descriptorilor globali, GOT *(Global Description Table),* contine descriptorii disponibili tuturor task-urilor din sistem (TI= 0):
  - descriptorii segmentelor de date si cod folosite de sistemul de operare;
  - descriptorii segmentelor de stare a taskurilor;
  - descriptorii pentru tabelele de descriptori locali ale sistemului.

  Adresa GOT este pastrata în registrul GOTR.
- tabelul deacriptorilor locali, LDT *(Local Description Table)* - contine descriptorii de segmente ale unui task dat: descriptorul segmentului de cod, date, stiva si al segmentului ce contine tabela vectorilor de întrerupere (TI = 1). Adresa LDT esle retinuta în registrul LDTR.
- tabelul vectorilor de întrerupere, IDT *(Interrupt Description Table)* contine descriptorii ce localizeaza rutinele de tratare a întreruperii, a caror adresa este pastrata în IDTR.

Index - indexul este un pointer de intrare în tabelul corespunzator de descriptori.

Un descriptor de segment dintr-un tabel accesat prin index, are urmatoarea structura (fig. 5.14):

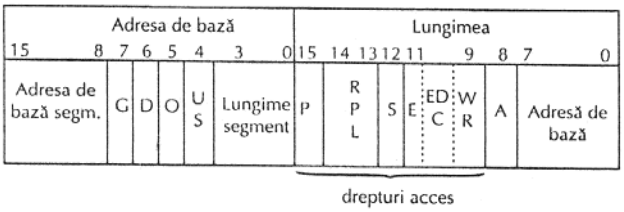


Fig. 5.14. Descriptor de segment

A *(Accessed)* = 0 - segmentul nu poate fl accesat
1 - selectorul de segment va fi încarcat în registrul segment

S *(Segment descriptor)* = 1 - descriptor segment de cod sau date
0 - descriptor segment de sistem sau întrerupere

E *(Executable)* = 1 - segment de cod
0 - alt segment decât cod

Daca S = 1, E = 0, atunci este un descriptor de segment de date.

ED *(Expansion Direction)*= 0 - extindere în sus, offset ≤ limita
1 - extindere în jos, offset > limita

W *(Wrlleable)* = 0 – scriere neautorizata
1 – scriere autorizata

Daca S = 1, E = 1, atunci este un descriptor de segment de cod:

C *(conforming)* = 1 - se executa daca are o prioritate ≤ RPL
0 - nu se executa deoarece prioritatea > RPL

R *(readable)*= 0 - nu autorizeaza citirea
1 - autorizeaza citirea

P *(present)* = 1 - segmentul se afla in memorie
0 - segmentul nu se afla în memorie

{U S} segment folosit de utilizator sau de sistemul de operare

O - pentru compatibilitatea cu procesoarele ulterioare

D *(dimension)* = 0 1 - segment pe 16 biti
1 - segment pe 32 biti

G *(granularity)* = 0 - lungimea segmentului este specificata în bytes
1 - lungime exprimata în pagini (1 pag. = 4 K).

(2) Din tabelul de descriptori segmente, se încarca automat registrul de descriptor segment (fig. 5.15), având structura:

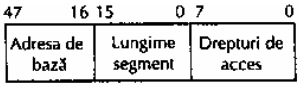


Fig. 5.15. Registru descriptor de segment

(3) Adresa de baza a segmentului, identifica segmentul din memoria interna,

(4) Adresa fizica se obtine relativ la adresa de baza prin intermediul sumatorului, la care se aduna offsetul comunicat prin magistrala de adrese de 24 biti ((I80286, I80386, SX $\Rightarrow 2^{24} = 2^{20} \cdot 2^4$ = 16 M RAM) sau 32 biti (ulterioare I80386 DX => $2^{32} = 2^{30}.2^2$ = 4 G RAM).

O alta modalitate de gestiune a memoriei pentru microprocesoarele pe 32 de biti, utilizata pentru sistemele de operare multitasking o constituie paginarea. Spre deosebire de segmentare, care modularizeaza programele si datele în segmente de lungime variabila, tinând seama de logica prelucrarii datelor si a modulelor ce alcatuiesc codul, paginarea divide datele si programele în zone de dimensiune fixa, numite pagini.

Tehnica paginarii (fig. 5.16) preia adresa liniara pe 32 de biti furnizata de magistrala de adrese, continând:

1) indexul unei intrari în directorul ce contine adresele tabelelor de pagini; pentru a identifica o adresa a unui tabel de pagini, acesta se va aduna cu adresa fizica de baza a directorului dc pagini (numarul de intrari este 2^{10} = 1024);

(2) adresa relativa a unui tabel de pagini (2^{10} - 1024 tabele), care se aduna la adresa de baza a tabelului determinat anterior pentru a obtine adresa paginii;

(3) offset-ul relativ la adresa paginii selectate anterior, care adunat la aceasta adresa, conduce la obtinerea adresei fizice pe 32 de biti.

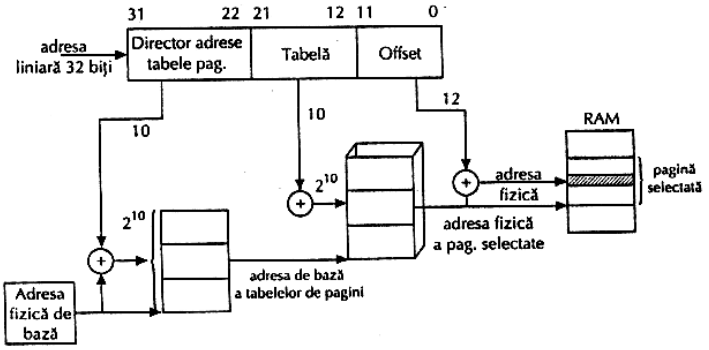


Fig. 5.16. Tehnica paginarii

5.1.7. Adresarea memoriei interne pe 32 de biti

La microprocesoarele pe 32 de biti, memoria interna nu mai este conectata la magistrala de 16 biti a sistemului, ci este atasata direct la magistrala locala a microprocesorului. Aceast fapt a extins modurile de adresare a rnernoriei interne la:

- adresarea complet liniara a 2^{32} = 4 G RAM (fig. 5.18);
- folosirea paginarii pentru a implernenta mecanismul de memorie virtuala prin care un bloc de memorie poate fi încarcat din memoria externa în mod automat, daca la momentul referirii nu se afla în memoria interna; în acest caz, un registru de segment va contine:

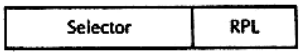


Fig. 5.17 Structura registrului de segment

Cu 14 biti se pot selecta $2^{14} = 2^{10} \cdot 2^4$ = 16 K adrese a câte 2^{32} off-seturi = $2^{14} \cdot 2^{32} = 2^{46}$ = 64 T spatiul de adrese virtuale.

- adresarea în mod real virtual, prin care microprocesorul poate simula comportamentul mai multor procesoare care lucreaza în mod real; deci orice utilizator sau task poate sa lucreze ca si când ar avea la dispozitie un intreg mediu real de 1 M.

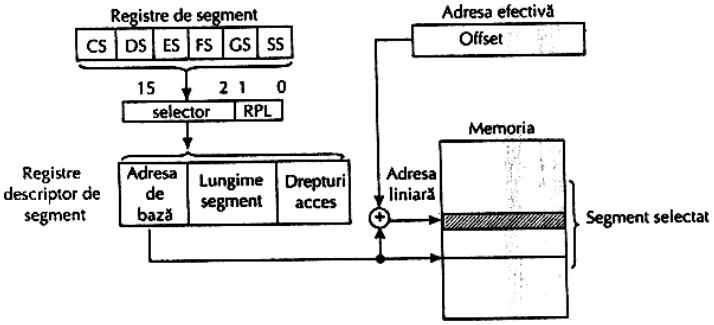


Fig. 5.18. Adresarea complet liniara

Modalitatile de adresare a memoriei la microprocesoarele pe 32 de biti sunt redate succint în figura 5.19.

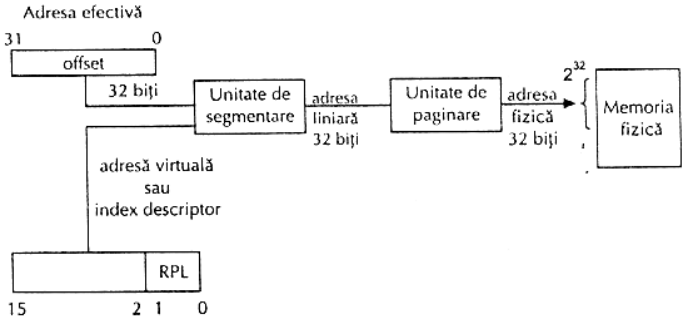


Fig. 5.19. Adresarea memoriei interne la microprocesoarele pe 32 de biti

Începând cu ultimele versiuni ale microprocesoarelor I80486 a fost introdus modul de operare SMM *(System Management Mode),* ce permite intrarea în oricare din modurile de adresare, fie prin software, fie prin îndeplinirea unor conditii hardware.

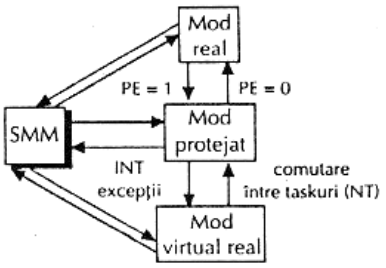


Fig. 5.20. Modul de operare SMM